Ecommerce Project

by Lokendersingh

Connected MySQL Workbench With Python

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import mysql.connector
import numpy as np
db = mysql.connector.connect(host='localhost',
username = 'root',
password = 'pass123',
database = 'ecommerce')


cur = db.cursor()
```

1. List all unique cities where customers are located.

```
query = """select distinct(customer_city) from customers """

cur.execute(query)

data = cur.fetchall()

data

df = pd.DataFrame(data)

df.head()
```

]

	0
0	franca
1	sao bernardo do campo
2	sao paulo
3	mogi das cruzes
4	campinas

2. Count the number of orders placed in 2017.

```
query = """select count(Order_id) from orders
where year(order_purchase_timestamp) = 2017 """

cur.execute(query)

data = cur.fetchall()

"Total Orders Placed in 2017 are", data[0][0]
```

```
('Total Orders Placed in 2017 are', 45101)
```

3. Find the total sales per category.

```
query = """select Upper(p.product_category) as category,
round(sum(pa.payment_value),2) as Sales from products p
join order_items o on p.product_id = o.product_id
join payments pa on o.order_id = pa.order_id
group by category; """

cur.execute(query)

data = cur.fetchall()

data

df= pd.DataFrame(data,columns=["Category","Sales"])

df
```

	Category	Sales
0	PERFUMERY	506738.66
1	FURNITURE DECORATION	1430176.39
2	TELEPHONY	486882.05
3	BED TABLE BATH	1712553.67
4	AUTOMOTIVE	852294.33
...	...	...
69	CDS MUSIC DVDS	1199.43
70	LA CUISINE	2913.53
71	FASHION CHILDREN'S CLOTHING	785.67
72	PC GAMER	2174.43
73	INSURANCE AND SERVICES	324.51

74 rows × 2 columns

4. Calculate the percentage of orders that were paid in installments.

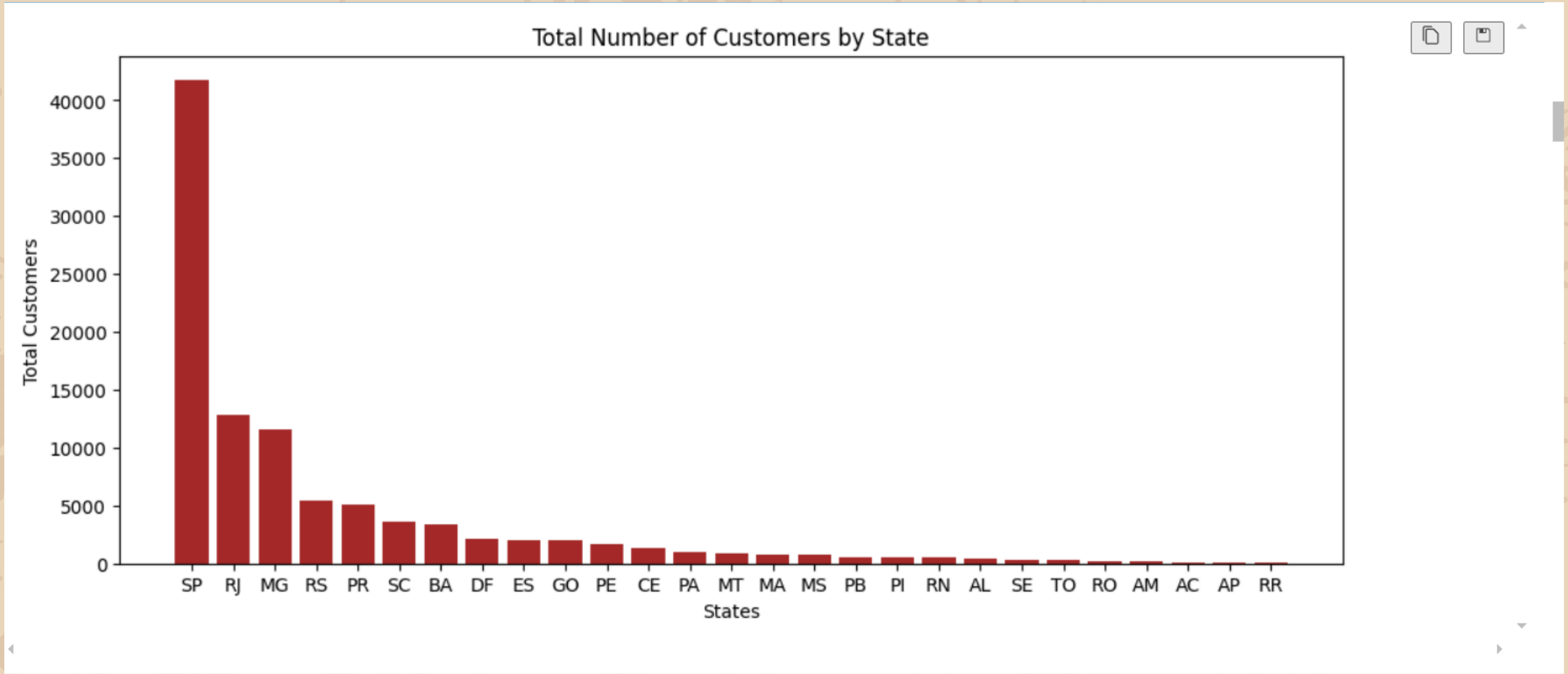
```python
query = """select sum(case when payment_installments >=1 then 1 else 0 end)/count(0)*100 as Percentage
from payments; """

cur.execute(query)

data = cur.fetchall()

"the percentage of orders that were paid in installments is",data[0][0]
```

Python

```
('the percentage of orders that were paid in installments is',
 Decimal('99.9981'))
```

5. Count the number of customers from each state.

```
query = """select customer_state,
count(customer_id) as total_count from customers
group by customer_state; """

cur.execute(query)

data = cur.fetchall()
df = pd.DataFrame(data,columns=["State","Total Customers"])
df= df.sort_values(by = "Total Customers", ascending= False)
df

plt.figure(figsize=(12, 5))
plt.xlabel("States")
plt.ylabel("Total Customers")
plt.bar(df["State"],df["Total Customers"],color="brown")
plt.title("Total Number of Customers by State")
plt.show()
```

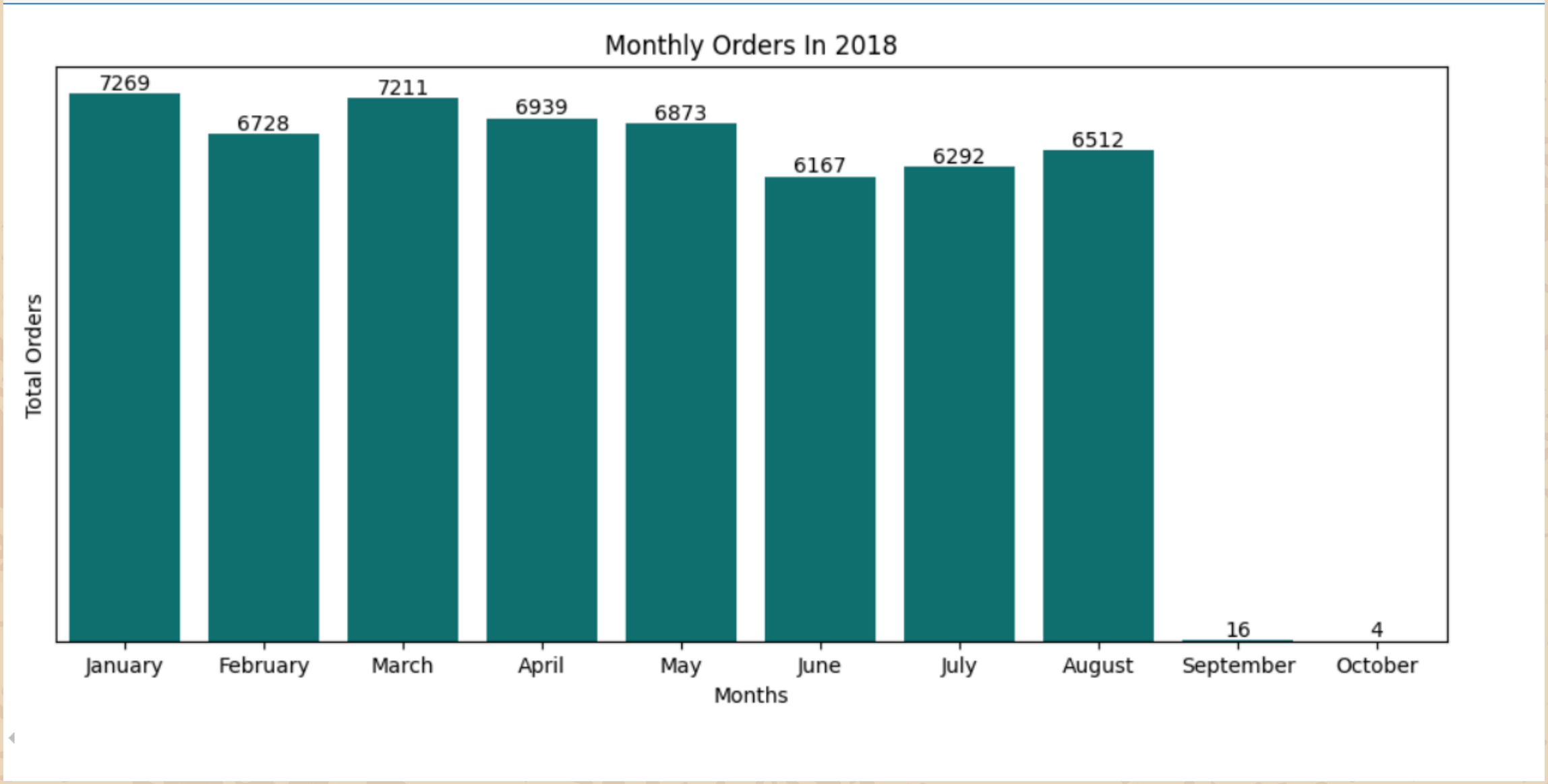

Total Number of Customers by State
Total Customers
0
5000
10000
15000
20000
25000
30000
35000
40000
SP
RJ
MG
RS
PR
SC
BA
DF
ES
GO
PE
CE
PA
MT
MA
MS
PB
PI
RN
AL
SE
TO
RO
AM
AC
AP
RR
States

6. Calculate the number of orders per month in 2018.

Generate | + Code | + Markdown

```
query = """select monthname(order_purchase_timestamp) as Month,
count(order_id) as "Total Orders" from orders
where year(order_purchase_timestamp) = 2018
group by Month ; """

cur.execute(query)

data = cur.fetchall()

data

df = pd.DataFrame(data,columns=["Months","Total Orders"])
df

o=["January","February","March","April","May","June","July","August","September","October"]

plt.figure(figsize=(12, 5))
ax=sns.barplot(x= df["Months"],y= df["Total Orders"],order = o,color="teal")
plt.xlabel("Months")
plt.ylabel("Total Orders")
plt.title("Monthly Orders In 2018")
ax.bar_label(ax.containers[0])
plt.yticks([])
plt.show()
```

Python


Monthly Orders In 2018
Total Orders
7269
6728
7211
6939
6873
6167
6292
6512
16
4
January
February
March
April
May
June
July
August
September
October
Months

7. Find the average number of products per order, grouped by customer city.

```
query = """with count_per_order as (select orders.order_id as order_id,orders.customer_id as customer_id,count(order_items.order_id) as order_counts
from orders join order_items
on orders.order_id = order_items.order_id
group by orders.order_id,orders.customer_id)

select customers.customer_city,round(avg(count_per_order.order_counts),2) as avg_orders from customers
join count_per_order on customers.customer_id = count_per_order.customer_id
group by customers.customer_city order by avg_orders desc; """

cur.execute(query)

data = cur.fetchall()

data

df =pd.DataFrame(data,columns=['Customer City','Avg Product/Order'])
df.head(10)
```

	Customer City	Avg Product/Order
0	padre carvalho	7.00
1	celso ramos	6.50
2	datas	6.00
3	candido godoi	6.00
4	matias olimpio	5.00
5	cidelandia	4.00
6	picarra	4.00
7	morro de sao paulo	4.00
8	teixeira soares	4.00
9	curralinho	4.00

8. Calculate the percentage of total revenue contributed by each product category.

```
query = """select p.product_category as category,
round((sum(pa.payment_value)/(select sum(payment_value) from payments))*100,2) as Sales_percentage
from products p
join order_items o on p.product_id = o.product_id
join payments pa on o.order_id = pa.order_id
group by category
order by Sales_percentage desc; """

cur.execute(query)

data = cur.fetchall()

data

df= pd.DataFrame(data,columns=["Category","Percentage Distribution"])

df.head(10)
```

	Category	Percentage Distribution
0	bed table bath	10.70
1	HEALTH BEAUTY	10.35
2	computer accessories	9.90
3	Furniture Decoration	8.93
4	Watches present	8.93
5	sport leisure	8.70
6	housewares	6.84
7	automotive	5.32
8	Garden tools	5.24
9	Cool Stuff	4.87

9. Identify the correlation between product price and the number of times a product has been purchased.

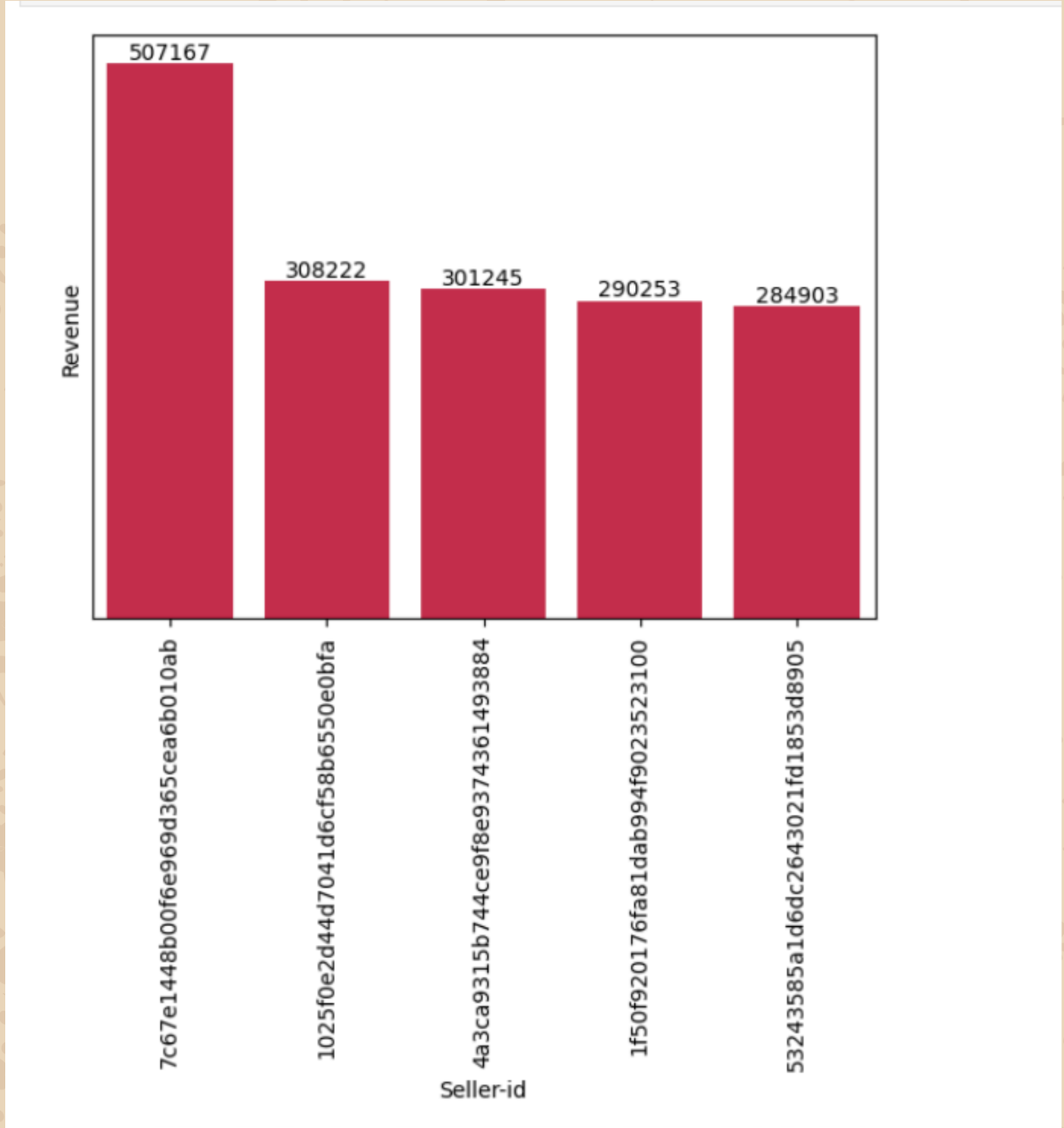
```
query = """select products.product_category as Category,
count(order_items.product_id) as "Total Counts",
round(avg(Order_items.price),2) as "Average Price"
from Products
join Order_items on products.product_id = Order_items.product_id
group by Category ;
"""


cur.execute(query)

data = cur.fetchall()

data

df= pd.DataFrame(data,columns=["Category","Total Orders","Average Price"])

df.head(10)

arr1 = df["Total Orders"]
arr2 = df["Average Price"]
a = np.corrcoef([arr1,arr2])
print("The Correlation is",a[0][-1])
```

```
The Correlation is -0.10631514167157562
```

10. Calculate the total revenue generated by each seller, and rank them by revenue.

```
query = """
select *,dense_rank() over(order by revenue desc) as "Rank" from
(select order_items.seller_id,sum(payments.payment_value) as Revenue from order_items
join payments on order_items.order_id = payments.order_id
group by order_items.seller_id) as a;
"""

cur.execute(query)

data = cur.fetchall()

data

df= pd.DataFrame(data,columns=["Seller-id","Revenue","Rank"])
df= df.head()
ax= sns.barplot(x=df["Seller-id"],y=df["Revenue"],color="crimson")
plt.xticks(rotation=90)
ax.bar_label(ax.containers[0])
plt.yticks([])
plt.show()
```

98]


507167
308222
301245
290253
284903
Revenue
7c67e1448b00f6e969d365cea6b010ab
1025f0e2d44d7041d6cf58b6550e0bfa
4a3ca9315b744ce9f8e9374361493884
1f50f920176fa81dab994f9023523100
53243585a1d6dc2643021fd1853d8905
Seller-id

11. Calculate the moving average of order values for each customer over their order history.

```
query = """
select customer_id,order_purchase_timestamp,payments,
avg(payments) over(partition by customer_id order by order_purchase_timestamp rows between 2 preceding and current row) as mov_avg from
(select orders.customer_id ,orders.order_purchase_timestamp,payments.payment_value as payments from payments
join orders on payments.order_id = orders.order_id) as a
"""

cur.execute(query)

data = cur.fetchall()

data

df= pd.DataFrame(data,columns=["Customer id","Timestamp","Payment","Moving Avg"])
df
```

	Customer id	Timestamp	Payment	Moving Avg
0	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.739998
1	000161a058600d5901f007fab4c27140	2017-07-16 09:40:32	67.41	67.410004
2	0001fd6190edaaf884bcaf3d49edf079	2017-02-28 11:06:43	195.42	195.419998
3	0002414f95344307404f0ace7a26f1d5	2017-08-16 13:09:20	179.35	179.350006
4	000379cdec625522490c315e70c7a9fb	2018-04-02 13:42:17	107.01	107.010002
...	...	...	...	...
103881	fffecc9f79fd8c764f843e9951b11341	2018-03-29 16:59:26	71.23	27.120001
103882	fffeda5b6d849fbd39689bb92087f431	2018-05-22 13:36:02	63.13	63.130001
103883	ffff42319e9b2d713724ae527742af25	2018-06-13 16:57:05	214.13	214.130005
103884	ffffa3172527f765de70084a7e53aae8	2017-09-02 11:53:32	45.50	45.500000
103885	fffffe8b65bbe3087b653a978c870db99	2017-09-29 14:07:03	18.37	18.370001

103886 rows × 4 columns

12. Calculate the cumulative sales per month for each year.

```
query = """
select years,months,payments,
sum(payments) over(order by years,months) as cum_Sales from
(select year(orders.order_purchase_timestamp) as years,
month(orders.order_purchase_timestamp) as months,
round(sum(payments.payment_value),2) as payments from orders
join payments on orders.order_id = payments.order_id
group by years,months
order by years,months) as a;
"""

cur.execute(query)

data = cur.fetchall()

data

df= pd.DataFrame(data,columns=["Years","Months","Payment","Cumulative sum"])
df
```

	Years	Months	Payment	Cumulative sum
0	2016	9	252.24	252.24
1	2016	10	59090.48	59342.72
2	2016	12	19.62	59362.34
3	2017	1	138488.04	197850.38
4	2017	2	291908.01	489758.39
5	2017	3	449863.60	939621.99
6	2017	4	417788.03	1357410.02
7	2017	5	592918.82	1950328.84
8	2017	6	511276.38	2461605.22
9	2017	7	592382.92	3053988.14
10	2017	8	674396.32	3728384.46
11	2017	9	727762.45	4456146.91
12	2017	10	779677.88	5235824.79
13	2017	11	1194882.80	6430707.59
14	2017	12	878401.48	7309109.07
15	2018	1	1115004.18	8424113.25
16	2018	2	992463.34	9416576.59
17	2018	3	1159652.12	10576228.71
18	2018	4	1160785.48	11737014.19
19	2018	5	1153982.15	12890996.34
20	2018	6	1023880.50	13914876.84
21	2018	7	1066540.75	14981417.59
22	2018	8	1022425.32	16003842.91
23	2018	9	4439.54	16008282.45
24	2018	10	589.67	16008872.12

13. Calculate the year-over-year growth rate of total sales.

```
query = """
with a as(select year(orders.order_purchase_timestamp) as years,
round(sum(payments.payment_value),2) as payments from orders
join payments on orders.order_id = payments.order_id
group by years
order by years)
select years,payments,lag(payments,1) over(order by years) as previous_year,
round(((payments-lag(payments,1) over(order by years))/lag(payments,1) over(order by years))*100,2) as "Growth rate" from a;
"""

cur.execute(query)

data = cur.fetchall()

data

df= pd.DataFrame(data,columns=["Years",'Sales','Previous_year','YOY Growth Rate %'])
df
```

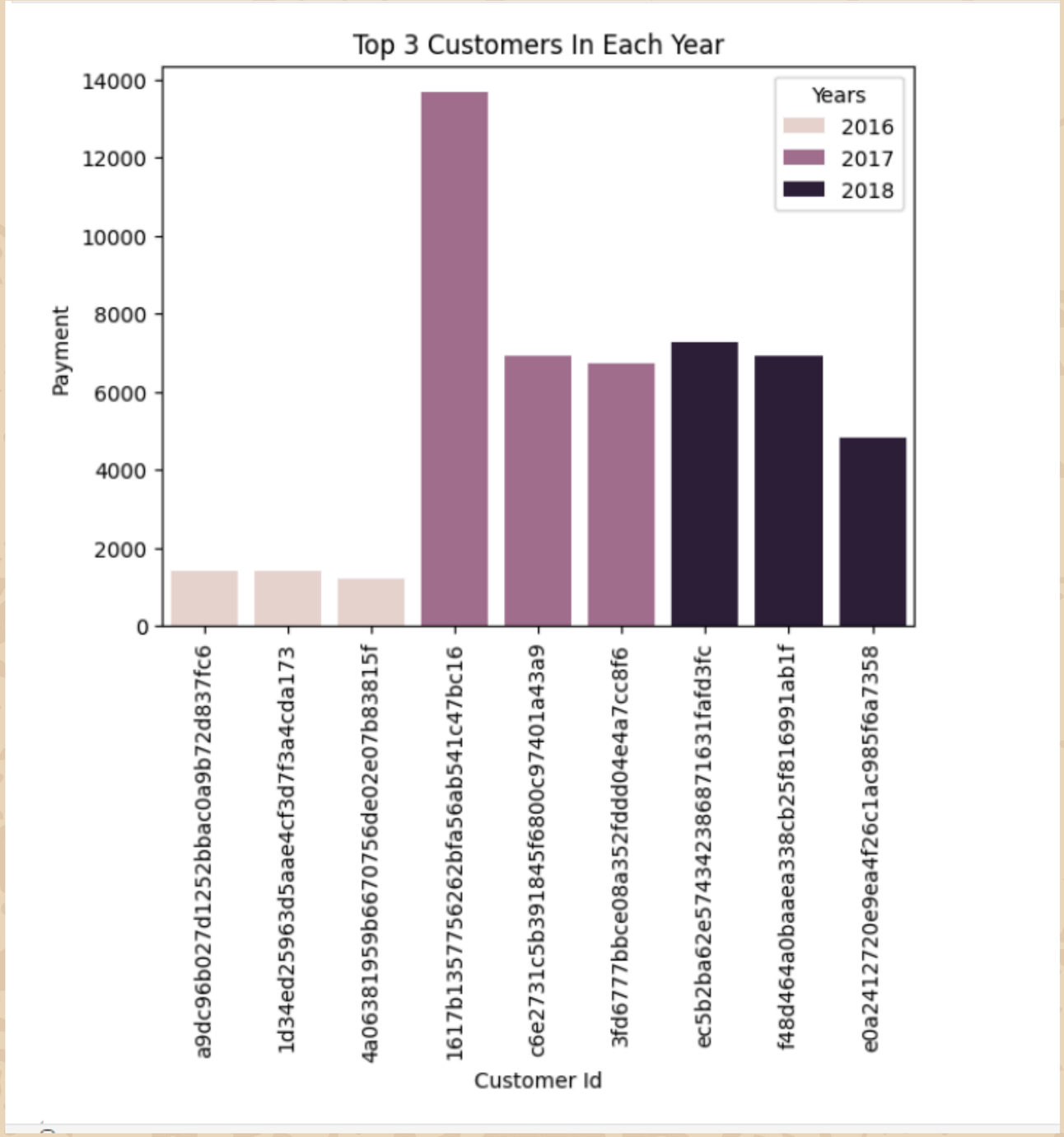
	Years	Sales	Previous_year	YOY Growth Rate %
0	2016	59362.34	NaN	NaN
1	2017	7249746.73	59362.34	12112.7
2	2018	8699763.05	7249746.73	20.0

14. Identify the top 3 customers who spent the most money in each year.

Generate | Code | Markdown

```
query = """
select years,customer_id,payment,d_rank from
(select year(orders.order_purchase_timestamp) as years,orders.customer_id,sum(payments.payment_value) as payment,
dense_rank() over (partition by year(orders.order_purchase_timestamp) order by sum(payments.payment_value) desc) d_rank
from orders join payments
on orders.order_id = payments.order_id
group by years,orders.customer_id) as a
where d_rank <=3;
"""

cur.execute(query)

data = cur.fetchall()

data

df= pd.DataFrame(data,columns=["Years","Customer Id",'Payment','Rank'])

sns.barplot(x= "Customer Id",y= "Payment", data = df, hue ='Years')
plt.title("Top 3 Customers In Each Year")
plt.xticks(rotation = 90)

plt.show()
```

19] ✓ 2.5s


Top 3 Customers In Each Year
Years
2016
2017
2018
Payment
14000
12000
10000
8000
6000
4000
2000
0
a9dc96b027d1252bbac0a9b72d837fc6
1d34ed25963d5aae4cf3d7f3a4cda173
4a06381959b6670756de02e07b83815f
1617b1357756262bfa56ab541c47bc16
c6e2731c5b391845f6800c97401a43a9
3fd6777bbce08a352fddd04e4a7cc8f6
ec5b2ba62e574342386871631fafd3fc
f48d464a0baaea338cb25f816991ab1f
e0a2412720e9ea4f26c1ac985f6a7358
Customer Id

Thank You